php

Search

date

Change language: French

Edit Report a Bug

(PHP 4, PHP 5)
date — Formate une date/heure locale

Description

```
string date ( string $format [, int $timestamp = time() ] )
```

Retourne une date sous forme d'une chaîne, au format donné par le paramètre **format**, fournie par le paramètre **timestamp** ou la date et l'heure courantes si aucun timestamp n'est fourni. En d'autres termes, le paramètre **timestamp** est optionnel et vaut par défaut la valeur de la fonction time().

Liste de paramètres

format

Le format de la date désirée. Voir les options de formatage ci-dessous. Il existe aussi de nombreuses constantes de dates qui peuvent être utilisées, ce qui fait que DATE_RSS va remplacer le format *"D, d M Y H:i:s"*.

Les caractères suivants sont reconnus dans le paramètre format

Caractères pour le paramètre format	Description	Exemple de valeurs retournées
Jour	---	---
d	Jour du mois, sur deux chiffres (avec un zéro initial)	*01* à *31*
D	Jour de la semaine, en trois lettres (et en anglais - par défaut : en anglais, ou sinon, dans la langue locale du serveur)	*Mon* à *Sun*
j	Jour du mois sans les zéros initiaux	*1* à *31*
l ('L' minuscule)	Jour de la semaine, textuel, version longue, en anglais	*Sunday* à *Saturday*
N	Représentation numérique ISO-8601 du jour de la semaine (ajouté en PHP 5.1.0)	*1* (pour Lundi) à *7* (pour Dimanche)
S	Suffixe ordinal d'un nombre pour le jour du mois, en anglais, sur deux lettres	*st*, *nd*, *rd* ou *th*. Fonctionne bien avec *j*
w	Jour de la semaine au format numérique	*0* (pour dimanche) à *6* (pour samedi)
z	Jour de l'année	*0* à *365*
Semaine	---	---
W	Numéro de semaine dans l'année ISO-8601, les semaines commencent le lundi (ajouté en PHP 4.1.0)	Exemple : *42* (la 42ème semaine de l'année)

Mois	---	---
F	Mois, textuel, version longue; en anglais, comme *January* ou *December*	*January* à *December*
m	Mois au format numérique, avec zéros initiaux	*01* à *12*
M	Mois, en trois lettres, en anglais	*Jan* à *Dec*
n	Mois sans les zéros initiaux	*1* à *12*
t	Nombre de jours dans le mois	*28* à *31*
Année	---	---
L	Est ce que l'année est bissextile	*1* si bissextile, *0* sinon.
o	L'année ISO-8601. C'est la même valeur que *Y*, excepté si le numéro de la semaine ISO (*W*) appartient à l'année précédente ou suivante, cette année sera utilisé à la place. (ajouté en PHP 5.1.0)	Exemples : *1999* ou *2003*
Y	Année sur 4 chiffres	Exemples : *1999* ou *2003*
y	Année sur 2 chiffres	Exemples : *99* ou *03*
Heure	---	---
a	Ante meridiem et Post meridiem en minuscules	*am* ou *pm*
A	Ante meridiem et Post meridiem en majuscules	*AM* ou *PM*
B	Heure Internet Swatch	*000* à *999*
g	Heure, au format 12h, sans les zéros initiaux	*1* à *12*
G	Heure, au format 24h, sans les zéros initiaux	*0* à *23*
h	Heure, au format 12h, avec les zéros initiaux	*01* à *12*
H	Heure, au format 24h, avec les zéros initiaux	*00* à *23*
i	Minutes avec les zéros initiaux	*00* à *59*
s	Secondes, avec zéros initiaux	*00* à *59*
u	Microsecondes (ajouté en PHP 5.2.2). Notez que la fonction **date()** génèrera toujours *000000* vu qu'elle prend un paramètre de type entier, alors que la méthode DateTime::format() supporte les microsecondes.	Exemple : *654321*
Fuseau horaire	---	---
e	L'identifiant du fuseau horaire (ajouté en PHP 5.1.0)	Exemples : *UTC*, *GMT*, *Atlantic/Azores*
I (i majuscule)	L'heure d'été est activée ou pas	*1* si oui, *0* sinon.
O	Différence d'heures avec l'heure de Greenwich (GMT), exprimée en heures	Exemple : *+0200*
P	Différence avec l'heure Greenwich (GMT) avec un deux-points entre les heures et les minutes (ajouté dans PHP 5.1.3)	Exemple : *+02:00*
T	Abréviation du fuseau horaire	Exemples : *EST*, *MDT* ...
Z	Décalage horaire en secondes. Le décalage des zones à l'ouest de la zone UTC est négative, et à l'est, il est positif.	*-43200* à *50400*
Date et Heure complète	---	---
c	Date au format ISO 8601 (ajouté en PHP 5)	2004-02-12T15:19:21+00:00
r	Format de date » RFC 2822	Exemple : *Thu, 21*

		Dec 2000 16:01:07 +0200
U	Secondes depuis l'époque Unix (1er Janvier 1970, 0h00 00s GMT)	Voir aussi time()

Les caractères non reconnus seront imprimés tels quels. "*Z*" retournera toujours *0* lorsqu'il est utilisé avec gmdate().

Note:
Sachant que cette fonction n'accepte que des entiers sous la forme de timestamp, le caractère *u* n'est utile que lors de l'utilisation de la fonction date_format() avec un timestamp utilisateur créé avec la fonction date_create().

timestamp

Le paramètre optionnel **timestamp** est un timestamp Unix de type entier qui vaut par défaut l'heure courante locale si le paramètre **timestamp** n'est pas fourni. En d'autres termes, il vaut par défaut la valeur de la fonction time().

Valeurs de retour

Retourne une date formatée. Si une valeur non-numérique est utilisée dans le paramètre **timestamp**, FALSE sera retourné et une erreur de niveau E_WARNING est émise.

Erreurs / Exceptions

Chaque appel à une fonction date/heure générera un message de type E_NOTICE si le fuseau horaire n'est pas valide, et/ou un message de type E_STRICT ou E_WARNING si vous utilisez la configuration du système ou la variable d'environnement `TZ`. Voir aussi date_default_timezone_set()

Historique

Version	Description
5.1.0	L'intervalle de validité d'un timestamp va généralement du Vendredi 13 Décembre 1901 20:45:54 GMT au Mardi 19 Janvier 2038 03:14:07 GMT. (Ces dates correspondent aux valeurs minimales et maximales des entiers 32 bits non-signés). Cependant, avant PHP 5.1.0, cette intervalle va du 01-01-1970 au 19-01-2038 sur quelques systèmes (e.g. Windows).
5.1.0	Émet un message de type E_STRICT et E_NOTICE lors d'erreurs de fuseaux horaires.
5.1.1	Il y a plusieurs constantes utiles de formats date/heure standards qui peuvent être utilisées pour spécifier le paramètre **format**.

Exemples

Exemple #1 Exemple avec date()

```
<?php
// Définit le fuseau horaire par défaut à utiliser. Disponible depuis PHP 5.1
date_default_timezone_set('UTC');
```

```
// Affichage de quelque chose comme : Monday
echo date("l");

// Affichage de quelque chose comme : Monday 8th of August 2005 03:12:46 PM
echo date('l jS \of F Y h:i:s A');

// Affiche : July 1, 2000 is on a Saturday
echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1, 2000));

/* utilise les constantes dans le paramètre format */
// Affichage de quelque chose comme : Wed, 25 Sep 2013 15:28:57 -0700
echo date(DATE_RFC2822);

// Affichage de quelque chose comme : 2000-07-01T00:00:00+00:00
echo date(DATE_ATOM, mktime(0, 0, 0, 7, 1, 2000));
?>
```

Vous pouvez faire afficher un caractère spécial dans la chaîne de format en le protégeant par un antislash. Si le caractère est lui-même une séquence incluant un antislash, vous devrez protéger aussi l'antislash.

Exemple #2 Protection des caractères dans la fonction date()

```
<?php
// Affichage de quelque chose comme : Wednesday the 15th
echo date('l \t\h\e jS');
?>
```

Il est possible d'utiliser **date()** et mktime() ensemble pour générer des dates dans le futur ou dans le passé.

Exemple #3 Exemple avec date() et mktime()

```
<?php
$tomorrow  = mktime(0, 0, 0, date("m")  , date("d")+1, date("Y"));
$lastmonth = mktime(0, 0, 0, date("m")-1, date("d"),   date("Y"));
$nextyear  = mktime(0, 0, 0, date("m"),   date("d"),   date("Y")+1);
?>
```

Note:
Cette méthode est plus sûre que simplement ajouter ou retrancher le nombre de secondes dans une journée ou un mois à un timestamp, à cause des heures d'hiver et d'été.

Voici maintenant quelques exemples de formatage avec **date()**. Notez que vous devriez échapper tous les autres caractères, car s'ils ont une signification spéciale, ils risquent de produire des effets secondaires indésirables. Notez aussi que les versions futures de PHP peuvent attribuer une signification à des lettres qui sont actuellement inertes. Lorsque vous échappez les caractères, pensez à utiliser des guillemets simples, pour que les séquences |*n* ne deviennent pas des nouvelles lignes.

Exemple #4 Exemple avec date()

```
<?php
// Aujourd'hui, le 12 Mars 2001, 5:16:18 pm, Fuseau horaire
// Mountain Standard Time (MST)

$today = date("F j, Y, g:i a");          // March 10, 2001, 5:16 pm
```

```
$today = date("m.d.y");                    // 03.10.01
$today = date("j, n, Y");                  // 10, 3, 2001
$today = date("Ymd");                      // 20010310
$today = date('h-i-s, j-m-y, it is w Day');      // 05-16-18, 10-03-01, 1631 1618 6 Satpm01
$today = date('\i\t \i\s \t\h\e jS \d\a\y.');    // It is the 10th day (10ème jour du mois).
$today = date("D M j G:i:s T Y");          // Sat Mar 10 17:16:18 MST 2001
$today = date('H:m:s \m \e\s\t\ \l\e\ \m\o\i\s'); // 17:03:18 m est le mois
$today = date("H:i:s");                    // 17:16:18
$today = date("Y-m-d H:i:s");              // 2001-03-10 17:16:18 (le format DATETIME de MySQL)
?>
```

Pour formater des dates dans d'autres langues, utilisez les fonctions setlocale() et strftime() au lieu de la fonction **date()**.

Notes

Note:
Pour générer un timestamp à partir d'une représentation de date, vous pouvez utiliser la fonction strtotime(). De plus, certaines bases de données disposent de fonctions pour convertir leurs propres formats de date en timestamp (par exemple, MySQL et sa fonction » UNIX_TIMESTAMP()).

Astuce Un timestamp représentant le début de la requête est disponible dans la variable `$_SERVER['REQUEST_TIME']` depuis PHP 5.1.

Voir aussi

- gmdate() - Formate une date/heure GMT/CUT
- idate() - Formate une date/heure locale en tant qu'entier
- getdate() - Retourne la date/heure
- getlastmod() - Retourne la date de dernière modification de la page
- mktime() - Retourne le timestamp UNIX d'une date
- strftime() - Formate une date/heure locale avec la configuration locale
- time() - Retourne le timestamp UNIX actuel
- strtotime() - Transforme un texte anglais en timestamp
- constantes de date

User Contributed Notes 39 notes

add a note

31 Jimmy 2 years ago

Things to be aware of when using week numbers with years.

```
<?php
echo date("YW", strtotime("2011-01-07")); // gives 201101
echo date("YW", strtotime("2011-12-31")); // gives 201152
echo date("YW", strtotime("2011-01-01")); // gives 201152 too
?>
```

BUT

```
<?php
```

```
echo date("oW", strtotime("2011-01-07")); // gives 201101
echo date("oW", strtotime("2011-12-31")); // gives 201152
echo date("oW", strtotime("2011-01-01")); // gives 201052 (Year is different than previous example)
?>
```

Reason:
Y is year from the date
o is ISO-8601 year number
W is ISO-8601 week number of year

Conclusion:
if using 'W' for the week number use 'o' for the year.

4 Anonymous 5 months ago

To quickly convert date("N") to a 0 based index with Sunday being represented as 0, you can run it against modulus 7:

```
<?php
$first_of_month_index = date('N', strtotime('4/1/1990')) % 7;
?>
```

11 FiraSEO 1 year ago

this how you make an HTML5 <time> tag correctly

```
<?php

echo '<time datetime="'.date('c').'">'.date('Y - m - d').'</time>';

?>
```

in the "datetime" attribute you should put a machine-readable value which represent time , the best value is a full time/date with ISO 8601 (date('c')) ,,, the attr will be hidden from users

and it doesn't really matter what you put as a shown value to the user,, any date/time format is okay !

This is very good for SEO especially search engines like Google .

1 Anonymous 2 months ago

It's common for us to overthink the complexity of date/time calculations and underthink the power and flexibility of PHP's built-in functions. Consider http://php.net/manual/en/function.date.php#108613

```
<?php
function get_time_string($seconds)
{
   return date('H:i:s', strtotime("2000-01-01 + $seconds SECONDS"));
}
```

8 SpikeDaCruz 8 years ago

The following function will return the date (on the Gregorian calendar) for Orthodox Easter (Pascha). Note that incorrect results will be returned for years less than 1601 or greater than 2399. This is because the Julian calendar (from which the Easter date is calculated) deviates from the Gregorian by one day for each century-year that is NOT a leap-year, i.e. the century is divisible by 4 but not by 10. (In the old Julian reckoning, EVERY 4th year was a leap-year.)

This algorithm was first proposed by the mathematician/physicist Gauss. Its complexity derives from the fact that the calculation is based on a combination of solar and lunar calendars.

```
<?php
function getOrthodoxEaster($date){
  /*
   Takes any Gregorian date and returns the Gregorian
   date of Orthodox Easter for that year.
  */
  $year = date("Y", $date);
  $r1 = $year % 19;
  $r2 = $year % 4;
  $r3 = $year % 7;
  $ra = 19 * $r1 + 16;
  $r4 = $ra % 30;
  $rb = 2 * $r2 + 4 * $r3 + 6 * $r4;
  $r5 = $rb % 7;
  $rc = $r4 + $r5;
  //Orthodox Easter for this year will fall $rc days after April 3
  return strtotime("3 April $year + $rc days");
}
?>
```

5 webmaster1989 at gmail dot com 1 year ago

Sometimes it is very useful to convert a sql timestamp to an also called NTP time. This is often used as time date notation in XML RSS pages. To convert a timestamp to this NTP notation try the following:

```
<?php
  echo date('D, d M Y h:i:s O', strtotime ($timestamp);
?>
```

4 nathan 2 years ago

```
<?php
/* the following variables are set to appropriate
  characters recognized by php version 5 that
  will get the date. To display the date, we have
  to use 'echo' or 'print' to send the variable
  data to the browser
*/

$day=date("l");
$date=date("j");
$suffix=date("S");
$month=date("F");
$year=date("Y");
echo $day . ", " . $month . " " . $date . $suffix . ", " . $year;
?>
```

rudimentary, simple way to due things, but it gets the job done for someone learning more on the subject.

2 Leopietroni 1 year ago

This function will add working day to a given timestamp

```
<?php
 function addworkinday($timestamp,$daystoadd){

    $dayoftheweek = date("N",$timestamp);
```

```
    $sum =$dayoftheweek +$daystoadd;

 while ($sum >= 6) {

    $daystoadd=$daystoadd+1;
   $sum=$sum-1;
}
 return $timestamp +(60*60*24*$daystoadd);

 }
?>
```

4 ghotinet 3 years ago

Most spreadsheet programs have a rather nice little built-in function called NETWORKDAYS to calculate the number of business days (i.e. Monday-Friday, excluding holidays) between any two given dates. I couldn't find a simple way to do that in PHP, so I threw this together. It replicates the functionality of OpenOffice's NETWORKDAYS function - you give it a start date, an end date, and an array of any holidays you want skipped, and it'll tell you the number of business days (inclusive of the start and end days!) between them.

I've tested it pretty strenuously but date arithmetic is complicated and there's always the possibility I missed something, so please feel free to check my math.

The function could certainly be made much more powerful, to allow you to set different days to be ignored (e.g. "skip all Fridays and Saturdays but include Sundays") or to set up dates that should always be skipped (e.g. "skip July 4th in any year, skip the first Monday in September in any year"). But that's a project for another time.

```
<?php

function networkdays($s, $e, $holidays = array()) {
   // If the start and end dates are given in the wrong order, flip them.
   if ($s > $e)
      return networkdays($e, $s, $holidays);

   // Find the ISO-8601 day of the week for the two dates.
   $sd = date("N", $s);
   $ed = date("N", $e);

   // Find the number of weeks between the dates.
   $w = floor(($e - $s)/(86400*7));    # Divide the difference in the two times by seven days to get the number of weeks.
   if ($ed >= $sd) { $w--; }        # If the end date falls on the same day of the week or a later day of the week than the start date, subtract
a week.

   // Calculate net working days.
   $nwd = max(6 - $sd, 0);    # If the start day is Saturday or Sunday, add zero, otherewise add six minus the weekday number.
   $nwd += min($ed, 5);    # If the end day is Saturday or Sunday, add five, otherwise add the weekday number.
   $nwd += $w * 5;        # Add five days for each week in between.

   // Iterate through the array of holidays. For each holiday between the start and end dates that isn't a Saturday or a Sunday, remove one
day.
   foreach ($holidays as $h) {
      $h = strtotime($h);
      if ($h > $s && $h < $e && date("N", $h) < 6)
         $nwd--;
   }

   return $nwd;
}
```

```
$start = strtotime("1 January 2010");
$end = strtotime("13 December 2010");

// Add as many holidays as desired.
$holidays = array();
$holidays[] = "4 July 2010";          // Falls on a Sunday; doesn't affect count
$holidays[] = "6 September 2010";       // Falls on a Monday; reduces count by one

echo networkdays($start, $end, $holidays);    // Returns 246

?>
```

Or, if you just want to know how many work days there are in any given year, here's a quick function for that one:

```
<?php

function workdaysinyear($y) {
   $j1 = mktime(0,0,0,1,1,$y);
   if (date("L", $j1)) {
      if (date("N", $j1) == 6)
         return 260;
      elseif (date("N", $j1) == 5 or date("N", $j1) == 7)
         return 261;
      else
         return 262;
   }
   else {
      if (date("N", $j1) == 6 or date("N", $j1) == 7)
         return 260;
      else
         return 261;
   }
}

?>
```

2 Tim Connolly 1 year ago

Here's my solution for looking up the month number by name (used when parsing an 'ls'):

```
<?php
 for($m=1;$m<=12;$m++){
   $month=date("M",mktime(0,0,0,$m,1,2000));
   $mon["$month"]=$m;
 }
?>
```

1 m_ocx at yahoo dot com 1 year ago

Here is a cool Date class to implement the date function:

```
<?php
/*
 * @author    Gchats
 *
 * Date class
 */
class Date
```

```
{
    private $shortDateFormat = "F j, Y";
    private $longDateFormat = "F j, Y, g:i a";
    private $timestamp = 0;

    /**
    * Default constructor
    *
    * @param    integer        $timestamp    unix time stamp
    */
    function __construct($timestamp = 0)
    {
        $this->timestamp = $timestamp;
    }

    /**
    * Returns the given timestamp in the constructor
    *
    * @return    integer        time stamp
    */
    public function getTime()
    {
        return (int) $this->timestamp;
    }

    /*
     * Returns long formatted date of the given timestamp
     *
     * @access public
     * @return     string    Long formatted date
     */
    public function long()
    {
        if ( $this->timestamp > 0 )
        {
            return date ( $this->longDateFormat , $this->timestamp );
        }
        else
        {
            return "";
        }
    }

    /*
     * Returns short formatted date of the given timestamp
     *
     * @access public
     * @return     string    Short formatted date
     */
    public function short()
    {
        if ( $this->timestamp > 0 )
        {
            return date ( $this->shortDateFormat , $this->timestamp );
        }
        else
        {
            return "";
```

```
        }
    }

    public function __toString()
    {
        return $this->timestamp;
    }

}
?>
```

3 Edward Rudd 4 years ago

To actually make use ot the "u" (microsecond) you need to use the DateTime object and not the date() function.

For example

```
<?php
$t = microtime(true);
$micro = sprintf("%06d",($t - floor($t)) * 1000000);
$d = new DateTime( date('Y-m-d H:i:s.'.$micro,$t) );

print $d->format("Y-m-d H:i:s.u");
?>
```

2 JonathanCross.com 5 years ago

```
<?php
// A demonstration of the new DateTime class for those
// trying to use dates before 1970 or after 2038.
?>
<h2>PHP 2038 date bug demo (php version <?php echo phpversion(); ?>)</h1>
<div style='float:left;margin-right:3em;'>
<h3>OLD Buggy date()</h3>
<?php
 $format='F j, Y';
 for ( $i = 1900; $i < 2050; $i++) {
  $datep = "$i-01-01";
  ?>
  Trying: <?php echo $datep; ?> = <?php echo date($format, strtotime($datep)); ?><br>
  <?php
 }
?></div>
<div style='float:left;'>
 <h3>NEW DateTime Class (v 5.2+)</h3><?php
 for ( $i = 1900; $i < 2050; $i++) {
  $datep = "$i-01-01";
  $date = new DateTime($datep);
  ?>
  Trying: <?php echo $datep; ?> = <?php echo $date->format($format); ?><br>
  <?php
 }
?></div>
```

2 Just.Kevin 4 years ago

In order to determine if a year is a leap year an earlier poster suggested simply checking to see if the year is a multiple of four:

```
<?php
function is_leapyear_broken($year = 2004) {
return ($year%4)==0;
}
?>
```

While this will work for the majority of years it will not work on years that are multiples of 100 but not multiples of 400 i.e.(2100).
A function not using php's date() function that will also account for this small anomaly in leap years:

```
<?php
function is_leapyear_working($year = 2004) {
   if((($year%4==0) && ($year%100!=0)) || $year%400==0) {
      return true;
   }
   return false;
}
?>
```

While is_leapyear_working will not return true for the few non-leap years divisible by four I couldn't tell you if this is more or less efficient than using php's date() as an even earlier poster suggested:

```
<?php
function is_leapyear($year = 2004) {
$is_leap = date('L', strtotime("$year-1-1"));
return $is_leap;
}
?>
```

1 Anonymous 11 months ago

Was trying to compare dates when I noticed that:

```
<?php

var_dump(date('d.m.Y', null));//string(10) "01.01.1970"
var_dump(date('d.m.Y', ''));//bool(false)

?>
```

Thought it's worth mentioning. Caused some weird logs to be produced in our system since this does not evaluate to the same.

2 Anon 2 years ago

I needed to convet a duration timestamp into H:i:s but whenever I did it kept bringing 5 back as 01:00:05 (due to some DST stuff) so I made this function to replace date(). It has no optimisations but hopefully someone might find it useful:

```
<?php
   function get_time_string(){
      $time = 3600+(60*32)+(50); // 01:32:50
      $time_string = '';

      $hours = (int)($time/(60*60));
      if(strlen($hours) > 1){
         $time_string = $hours.':';
      }else{
         $time_string = '0'.$hours.':';
      }
```

```
    $minutes = (int)(($time%(60*60))/(60));
    if($minutes >= 1){
      if(strlen($minutes) > 1){
        $time_string .= $minutes.':';
      }else{
        $time_string .= '0'.$minutes.':';
      }

      $seconds = ($time%(60*60))%(60);
      if(strlen($seconds) > 1){
        $time_string .= $seconds;
      }else{
        $time_string .= '0'.$seconds;
      }
    }else{
      if(strlen($time) > 1){
        $time_string .= '00:'.$time;
      }else{
        $time_string .= '00:0'.$time;
      }
    }
    return $time_string;
  }
?>
```

1 stokestack at gmail dot com 1 year ago

If you want to find your server's timezone offset from GMT, it seems as though you could just do:

date('Z')

to get the number of seconds offset. But PHP requires that you call date_default_timezone_set(). So if you have to hard-code a timezone, why not simply hard-code a variable that tells you the offset from GMT? If you set the timezone to GMT, the dates in your database will still be in local time, but time('Z') will return zero.

To keep your code portable across servers in different timezones, you can do this:

date_default_timezone_set(date_default_timezone_get())

This keeps PHP from complaining that you haven't called date_default_timezone_set(), but makes your code portable. Ridiculous.

1 Bas Vijfwinkel 2 years ago

Note that some formatting options are different from MySQL.
For example using a 24 hour notation without leading zeros is the option '%G' in PHP but '%k' in MySQL.
When using dynamically generated date formatting string, be careful to generate the correct options for either PHP or MySQL.

2 bakerj417 at gmail dot com 2 years ago

If you are having an issue getting u to work so is everyone else. The solution that I am using which I found on another site(so not taking credit) is to use this:

date("Y/m/d H:i:s"). substr((string)microtime(), 1, 6);

that will give you:

yyyy/mm/dd hh:ii:ss.uuuuuu

hope this helps someone in need!

thanks all

3 jc 6 years ago

date("W") returns the iso8601 week number, while date("Y") returns the _current_ year. This can lead to odd results. For example today (dec 31, 2007) it returns 1 for the week and of course 2007 for the year. This is not wrong in a strict sense because iso defines this week as the first of 2008 while we still have 2007.

So, if you don't have another way to safely retrieve the year according to the iso8061 week-date - strftime("%G") doesn't work on some systems -, you should be careful when working with date("W").

For most cases strftime("%W") should be a safe replacement.

[edit: Much easier is to use "o" (lower case O) instead of "Y"]

3 mel dot boyce at gmail dot com 8 years ago

I've been flicking through the comments looking for some succinct date code and have noticed an alarming number of questions and over-burdened examples related to date mathematics. One of the most useful skills you can utilize when performing date math is taking full advantage of the UNIX timestamp. The UNIX timestamp was built for this kind of work.

An example of this relates to a comment made by james at bandit-dot-co-dot-en-zed. James was looking for a way to calculate the number of days which have passed since a certain date. Rather than using mktime() and a loop, James can subtract the current timestamp from the timestamp of the date in question and divide that by the number of seconds in a day:

```
<?php
$days = floor((time() - strtotime("01-Jan-2006"))/86400);
print("$days days have passed.\n");
?>
```

Another usage could find itself in a class submitted by Kyle M Hall which aids in the creation of timestamps from the recent past for use with MySQL. Rather than the looping and fine tuning of a date, Kyle can use the raw UNIX timestamps (this is untested code):

```
<?php
$ago = 14; // days
$timestamp = time() - ($ago * 86400);
?>
```

Hopefully these two examples of "UNIX-style" timestamp usage will help those finding date mathematics more elusive than it should be.

1 eduardo at digmotor dot com dot br 5 years ago

Thanks to tcasparr at gmail dot com for the great idea (at least for me) ;)
I changed the code a little to replicate the functionality of date_parse_from_format, once I don't have PHP 5.3.0 yet. This might be useful for someone. Hope you don't mind changing your code tcasparr at gmail dot com.

```
<?php
/*******************************************************
 * Simple function to take in a date format and return array of associated
 * formats for each date element
 *
 * @return array
 * @param string $strFormat
 *
 * Example: Y/m/d g:i:s becomes
 * Array
 * (
 *     [year] => Y
```

```
 *     [month] => m
 *     [day] => d
 *     [hour] => g
 *     [minute] => i
 *     [second] => s
 * )
 *
 *  This function is needed for  PHP < 5.3.0
 *******************************************************/
function dateParseFromFormat($stFormat, $stData)
{
   $aDataRet = array();
   $aPieces = split('[:/.\ \-]', $stFormat);
   $aDatePart = split('[:/.\ \-]', $stData);
   foreach($aPieces as $key=>$chPiece)
   {
      switch ($chPiece)
      {
         case 'd':
         case 'j':
            $aDataRet['day'] = $aDatePart[$key];
            break;

         case 'F':
         case 'M':
         case 'm':
         case 'n':
            $aDataRet['month'] = $aDatePart[$key];
            break;

         case 'o':
         case 'Y':
         case 'y':
            $aDataRet['year'] = $aDatePart[$key];
            break;

         case 'g':
         case 'G':
         case 'h':
         case 'H':
            $aDataRet['hour'] = $aDatePart[$key];
            break;

         case 'i':
            $aDataRet['minute'] = $aDatePart[$key];
            break;

         case 's':
            $aDataRet['second'] = $aDatePart[$key];
            break;
      }

   }
   return $aDataRet;
}
?>
```

Also, if you need to change the format of dates:

```
<?php
function changeDateFormat($stDate,$stFormatFrom,$stFormatTo)
{
  // When PHP 5.3.0 becomes available to me
  //$date = date_parse_from_format($stFormatFrom,$stDate);
  //For now I use the function above
  $date = dateParseFromFormat($stFormatFrom,$stDate);
  return date($stFormatTo,mktime($date['hour'],
                                 $date['minute'],
                                 $date['second'],
                                 $date['month'],
                                 $date['day'],
                                 $date['year']));
}

?>
```

1 gerben at gerbenwijnja dot nl 2 years ago

I use the function below to calculate the Unix timestamp of the start of a week. It includes a boolean flag to request a GMT offset instead of the current locale setting.

```
<?php

function getWeekOffsetTimestamp($year, $week, $useGmt = false) {
	if ($useGmt) {
		// Backup timezone and set to GMT
		$timezoneSettingBackup = date_default_timezone_get();
		date_default_timezone_set("GMT");
	}

	// According to ISO-8601, January 4th is always in week 1
	$halfwayTheWeek = strtotime($year."0104 +".($week - 1)." weeks");

	// Subtract days to Monday
	$dayOfTheWeek = date("N", $halfwayTheWeek);
	$daysToSubtract = $dayOfTheWeek - 1;

	// Calculate the week's timestamp
	$unixTimestamp = strtotime("-$daysToSubtract day", $halfwayTheWeek);

	if ($useGmt) {
		// Reset timezone to backup
		date_default_timezone_set($timezoneSettingBackup);
	}

	return $unixTimestamp;
}

?>
```

2 matthew dot hotchen at worldfirst dot com 3 months ago

FYI: there's a list of constants with predefined formats on the DateTime object, for example instead of outputting ISO 8601 dates with:

```
<?php
echo date('c');
```

```
?>
```

or

```
<?php
echo date('Y-m-d\TH:i:sO');
?>
```

You can use

```
<?php
echo date(DateTime::ISO8601);
?>
```

instead, which is much easier to read.

1 lb at bostontech dot net 4 years ago

Not sure why this got ignored the first time, but this is an even simpler way to check leap year:

```
<?php
function isLeapYear($year)
   { return ((($year%4==0) && ($year%100)) || $year%400==0) ? (true):(false); }
?>
```

1 Manu Manjunath 1 month ago

If you want to use "u" format specifier for micrseconds without changing to DateTime object, you may write a function as below:

```
<?php
/**
 * Quick replacement to date() function to handle the 'u' format specifier (for microseconds)
 * @param string $format Date format string - the same format string you would pass to date() function
 * @param float $timestamp [optional] Unix timestamp with microseconds - Typically output of <b>microtime(true)</b>
 * @return string Formatted string
 */
function date_with_micro($format, $timestamp = null) {
   if (is_null($timestamp) || $timestamp === false) {
      $timestamp = microtime(true);
   }
   $timestamp_int = (int) floor($timestamp);
   $microseconds = (int) round(($timestamp - floor($timestamp)) * 1000000.0, 0);
   $format_with_micro = str_replace("u", $microseconds, $format);
   return date($format_with_micro, $timestamp_int);
}
?>
```

You can safely replace your date() function with date_with_micro().

0 ttt_joe_08 3 months ago

Just FYI, it's more appropriate to say "UTC", not "GMT". GMT was given up in 1972 and UTC is now the proper way. The reason being G stands for Greenwich, which naturally upset some people.

0 adityabhai at gmail dot com 7 months ago

For Microseconds, we can get by following:

```
echo date('Ymd His'.substr((string)microtime(), 1, 8).' e');
```

Thought, it might be useful to someone !

-1 Anonymous 5 years ago

Correct format for a MySQL DATETIME column is

```
<?php $mysqltime = date ("Y-m-d H:i:s", $phptime); ?>
```

-1 scott at keenot dot es 11 months ago

If anyone needs a really fast function for converting a datetime string (i.e. as retrieved from a MySQL DATETIME entry) into a human-friendly time output analogous to date($format, $time), here's a useful function.

```
<?php
function fdate($datetimestring = '1970-01-01 00:00:00', $format = 'U') {
  // Create a datetime object, return it formatted
  // If you want to give credit for this somewhere, thanks.
  // You really don't have to though; this is kinda obvious
  $dt = new DateTime($datetimestring);
  return $dt->format($format);
}
?>
```

The main purpose of this is to reduce lines of code and allow inline coding. For example:

```
<?php
/* ... */
echo "This page was submitted on ".fdate($row['created'], 'F j, Y g:i:s A')." and last modified ".fdate($row['modified'], 'F j, Y g:i:s A')."<br />\n";
/* ... */
?>
```

-1 @PeteWilliams 3 years ago

If you want to use HTML5's <date> tag, the following code will generate the machine-readable value for the 'datetime' attribute:

```
<?php

/**
 * formats the date passed into format required by 'datetime' attribute of <date> tag
 * if no intDate supplied, uses current date.
 * @param intDate integer optional
 * @return string
 **/
function getDateTimeValue( $intDate = null ) {

    $strFormat = 'Y-m-d\TH:i:s.uP';
    $strDate = $intDate ? date( $strFormat, $intDate ) : date( $strFormat ) ;

    return $strDate;
}

echo getDateTimeValue();

?>
```

-2 frank at interactinet dot com 2 years ago

If you want to compare this week with the same week last year, here is some code to get you the time at the beginning of the week. You

can then add days, hours, etc to get to the day of the week that you want to know about.

```
<?php
    $time_passed = (date('N')-1)* 24 * 3600; // time since start of week in days
    $startOfWeek = mktime(0,0,0,date('m'),date('d'),date('Y')) - $time_passed;


    $lastyear = $startOfWeek - 365*24*3600;

    // make sure time used from last year is the same week of the year
    $weekdiff = date('W') - date('W',$lastyear);
    if($weekdiff != 0)
    {
      $lastyear = $lastyear + ($weekdiff*7*24*3600);
    }

    $lastyear_time_passed = (date('N',$lastyear)-1) * 24 * 3600; // time since start of week in days

    $startOfWeek_lastyear = mktime(0,0,0,date('m',$lastyear),date('d',$lastyear),date('Y',$lastyear)) - $lastyear_time_passed;
?>
```

So now you have the unix time for the start of this week ($startOfWeek), and the start of the same week last year ($startOfWeek_lastyear).

You can convert back to datetime format easily:

```
<?php
    echo date('Y-m-d H:i:s',$startOfWeek).'<br>';
    echo date('Y-m-d H:i:s',$startOfWeek_lastyear).'<br><br>';

    echo date('l F jS, Y',$startOfWeek).'<br>';
    echo date('l F jS, Y',$startOfWeek_lastyear);
?>
```

-4 chubby at chicks dot com 5 years ago

```
<?php
/**
  * Checks wether a date is between an interval
  *
  * Usage:
  *
  * // check if today is older than 2008/12/31
  * var_dump(currentDayIsInInterval('2008/12/31'));
  * // check if today is younger than 2008/12/31
  * var_dump(currentDayIsInInterval(null,'2008/12/31'));
  * // check if today is between 2008/12/01 and 2008/12/31
  * var_dump(currentDayIsInInterval('2008/12/01','2008/12/31'));
  *
  * Will trigger errors if date is in wrong format, notices if $begin > $end
  *
  * @param string $begin Date string as YYYY/mm/dd
  * @param string $end Date string as YYYY/mm/dd
  * @return bool
  */
function currentDayIsInInterval($begin = '',$end = '')
{
    $preg_exp = '"[0-9][0-9][0-9][0-9]/[0-9][0-9]/[0-9][0-9]"';
    $preg_error = 'Wrong parameter passed to function '.__FUNCTION__.' : Invalide date
```

```
format. Please use YYYY/mm/dd.';
	$interval_error = 'First parameter in '.__FUNCTION__.' should be smaller than
second.';
	if(empty($begin))
	{
		$begin = 0;
	}
	else
	{
		if(preg_match($preg_exp,$begin))
		{
			$begin = (int)str_replace('/','',$begin);
		}
		else
		{
			trigger_error($preg_error,E_USER_ERROR);
		}
	}
	if(empty($end))
	{
		$end = 99999999;
	}
	else
	{
		if(preg_match($preg_exp,$end))
		{
			$end = (int)str_replace('/','',$end);
		}
		else
		{
			trigger_error($preg_error,E_USER_ERROR);
		}
	}
	if($end < $begin)
	{
		trigger_error($interval_error,E_USER_WARNING);
	}
	$time = time();
	$now = (int)(date('Y',$time).date('m',$time).date('j',$time));
	if($now > $end or $now < $begin)
	{
		return false;
	}
	return true;
}
?>
```

-4 lehal2 at hotmail dot com 1 year ago

here is an example how you can make numeric days of the week from 1 to 7(Monday to Friday)

```
<?php
$currentdate  = mktime(0, 0, 0, date("m")  , date("d"), date("Y"));
    echo $day_eg1 = date ('N',$currentdate);
     echo $day_eg2 = date("N", $today+1 * 24 * 3600);
   echo $day_eg3= date("N", $today+2 * 24 * 3600);
   echo $day_eg4 = date("N", $today+3 * 24 * 3600);
   echo $day_eg5 = date("N", $today+4 * 24 * 3600);
```

```
    echo $day_eg6 = date("N", $today+5 * 24 * 3600);
    echo $day_eg7 = date("N", $today+6 * 24 * 3600);
?>
```

-4 Chris 1 year ago

Use this to convert the local/UTC hour to the UTC/local hour:

```
<?php
for($utc_to_local = array(), $offset = date('Z'), $h = 0; $h < 24; $utc_to_local[] = date('G', mktime($h++)+$offset));
$local_to_utc = array_flip($utc_to_local);

echo "2 am local is ", $local_to_utc[2], " UTC";
echo "3 pm UTC is ", $utc_to_local[15], " local";
?>
```

This is useful when you need to do many conversions. Lookup tables are faster than calling date() and mktime() multiple times.

-2 Jacques Marais 3 months ago

If you want to print something like: ☐Tuesday, the 14th of January, 2014

Use this:

```
<?php
echo date("l", strtotime("now")).', the'.date(" jS", strtotime("now")).' of'.date(" F, Y", strtotime("now"));
?>
```

This is because you cannot use words in the date string. If you use words in the date string it will be seen as a format character

So if you use:

```
<?php
echo date("l, the jS of F, Y", strtotime("now"));
?>
```

It will print something like: Tuesday, 3108Europe/Berlin 14th 2014f January, 2014

-9 Anonymous 1 year ago

To find last sunday for given date

```
<?php
        $day = '2012-10-04';
        echo 'last sunday :  '.date("Y-m-d",strtotime($day." last Sunday "));
?>
```

output:

last sunday : 2012-09-30

-6 blinov vyacheslav AT gmail.com 2 years ago

It was oblivious and discouraging that it dont mentioned in docs. If you will use W to get week number be aware:
first days of year can be in a week of previous year, and week number always has leading zero

```
<?php
```

```
echo date("YW", strtotime("2011-01-07")); // gives 201101
echo date("YW", strtotime("2011-01-01")); // gives 201152
echo date("YW", strtotime("2011-12-31")); // gives 201152 too

?>
```

so you can`t rely on number of week given from this function inside your program if you want to use it for some logic

-15 matt 2 years ago

date() has some strange behavior at extremely high values:

```
<?php
echo "9223372036854775805: " . date("Y-m-d g:i:s a", 9223372036854775805) . "\n";
echo "9223372036854775806: " . date("Y-m-d g:i:s a", 9223372036854775806) . "\n";
echo "9223372036854775807: " . date("Y-m-d g:i:s a", 9223372036854775807) . " (0x7FFFFFFFFFFFFFFF)\n";
echo "9223372036854775808: " . date("Y-m-d g:i:s a", 9223372036854775808) . "\n";
echo "9223372036854775809: " . date("Y-m-d g:i:s a", 9223372036854775809) . "\n";
echo "9223372036854775810: " . date("Y-m-d g:i:s a", 9223372036854775810) . "\n";
echo "...\n";
echo "9223372036854776832: " . date("Y-m-d g:i:s a", 9223372036854776832) . "\n";
echo "9223372036854776833: " . date("Y-m-d g:i:s a", 9223372036854776833) . "\n";
echo "...\n";
echo "9223372036854778879: " . date("Y-m-d g:i:s a", 9223372036854778879) . "\n";
echo "9223372036854778880: " . date("Y-m-d g:i:s a", 9223372036854778880) . "\n";
echo "...\n";
echo "9223372036854780928: " . date("Y-m-d g:i:s a", 9223372036854780928) . "\n";
echo "9223372036854780929: " . date("Y-m-d g:i:s a", 9223372036854780929) . "\n";
echo "...\n";
echo "9223372036854782975: " . date("Y-m-d g:i:s a", 9223372036854782975) . "\n";
echo "9223372036854782976: " . date("Y-m-d g:i:s a", 9223372036854782976) . "\n";
echo "...\n";
echo "9223372036854785024: " . date("Y-m-d g:i:s a", 9223372036854785024) . "\n";
echo "9223372036854785025: " . date("Y-m-d g:i:s a", 9223372036854785025) . "\n";
echo "...\n";
echo "9223372036854787071: " . date("Y-m-d g:i:s a", 9223372036854787071) . "\n";
echo "9223372036854787072: " . date("Y-m-d g:i:s a", 9223372036854787072) . "\n";
echo "...\n";
echo "9223372036854789120: " . date("Y-m-d g:i:s a", 9223372036854789120) . "\n";
echo "9223372036854789121: " . date("Y-m-d g:i:s a", 9223372036854789121) . "\n";
echo "...\n";
echo "9223372036854791167: " . date("Y-m-d g:i:s a", 9223372036854791167) . "\n";
echo "9223372036854791168: " . date("Y-m-d g:i:s a", 9223372036854791168) . "\n";
echo "...\n";
echo "9223372036854793215: " . date("Y-m-d g:i:s a", 9223372036854793215) . "\n";
echo "9223372036854793216: " . date("Y-m-d g:i:s a", 9223372036854793216) . "\n";
echo "9223372036854793217: " . date("Y-m-d g:i:s a", 9223372036854793217) . "\n";
echo "9223372036854793218: " . date("Y-m-d g:i:s a", 9223372036854793218) . "\n";
?>
```

Output:

```
9223372036854775805: 292277026596-12-04 10:30:05 am
9223372036854775806: 292277026596-12-04 10:30:06 am
9223372036854775807: 292277026596-12-04 10:30:07 am (0x7FFFFFFFFFFFFFFF)
9223372036854775808: 292277026596-12-04 10:30:08 am
9223372036854775809: 292277026596-12-04 10:30:08 am
9223372036854775810: 292277026596-12-04 10:30:08 am
```

...
9223372036854778879: 292277026596-12-04 10:30:08 am
9223372036854778880: 292277026596-12-04 11:04:16 am
...
9223372036854778879: 292277026596-12-04 11:04:16 am
9223372036854778880: 292277026596-12-04 11:38:24 am
...
9223372036854780928: 292277026596-12-04 11:38:24 am
9223372036854780929: 292277026596-12-04 12:12:32 pm
...
9223372036854782975: 292277026596-12-04 12:12:32 pm
9223372036854782976: 292277026596-12-04 12:46:40 pm
...
9223372036854785024: 292277026596-12-04 12:46:40 pm
9223372036854785025: 292277026596-12-04 1:20:48 pm
...
9223372036854787071: 292277026596-12-04 1:20:48 pm
9223372036854787072: 292277026596-12-04 1:54:56 pm
...
9223372036854789120: 292277026596-12-04 1:54:56 pm
9223372036854789121: 292277026596-12-04 2:29:04 pm
...
9223372036854791167: 292277026596-12-04 2:29:04 pm
9223372036854791168: 292277026596-12-04 3:03:12 pm
...
9223372036854793215: 292277026596-12-04 3:03:12 pm
9223372036854793216: 292277026596-12-04 3:03:12 pm
9223372036854793217: -292277022657-01-27 8:37:04 am
9223372036854793218: -292277022657-01-27 8:37:04 am

So, the last reliable unix timecode is 9223372036854775808 (0x1000000000000000). Not that you would probably ever need a date that high.

add a note

Fonctions Date/Heure

checkdate
date_add
date_create_from_format
date_create_immutable_from_format
date_create_immutable
date_create
date_date_set
date_default_timezone_get
date_default_timezone_set
date_diff
date_format
date_get_last_errors
date_interval_create_from_date_string
date_interval_format
date_isodate_set
date_modify
date_offset_get
date_parse_from_format

date_parse
date_sub
date_sun_info
date_sunrise
date_sunset
date_time_set
date_timestamp_get
date_timestamp_set
date_timezone_get
date_timezone_set
» **date**
getdate
gettimeofday
gmdate
gmmktime
gmstrftime
idate
localtime
microtime
mktime
strftime
strptime
strtotime
time
timezone_abbreviations_list
timezone_identifiers_list
timezone_location_get
timezone_name_from_abbr
timezone_name_get
timezone_offset_get
timezone_open
timezone_transitions_get
timezone_version_get

Copyright © 2001-2014 The PHP Group My PHP.net Contact Other PHP.net sites Mirror sites

Privacy policy